Implementació d'un workflow d'execució paral·lela mitjançant el framework REANA

Oriol Martínez Acón

Resum– Els investigadors requereixen poder parametritzar i estructurar l'anàlisi de manera que puguin ser reproduïbles perquè la comunitat científica ho validi. És per això que institucions i diversos organismes estan fent grans esforços per a garantir la reproductibilitat dels seus estudis. El present treball explora el framework REANA i l'ús de contenidors computacionals com a eines de modelització, parametrització i preservació de workflows científics. Com a cas d'ús s'implementa un workflow de conversió de fitxers SuperStar a format DL3 aplicat al projecte MAGIC. A partir de la modelització del workflow es proposa una estratègia d'implementació de paral·lelisme a les etapes de major temps de còmput, millorant el rendiment amb un Speed-Up de 3.14x en les etapes crítiques de menor ús de disc dur. L'accés a disc resulta un factor limitant que no permet la millora del rendiment d'altres etapes, es proposen solucions per a superar les limitacions de recursos.

Paraules clau– Reproduible, Anàlisis de dades, ReAna, Kubernetes, Docker, Contenidors, Rancher, Workflows, Common Workflow Language, DL3, MAGIC, Paral·lelisme.

Abstract– Researchers need to be able to parameterize and structure the analysis in ways that can be reproducible to be validated by the scientific community. This is why there are institutions and various bodies that make efforts to guarantee the reproducibility of their studies. Current work explores the REANA framework and the use of computational containers as tools for modeling, parameterizing and conserving scientific workflows. As a use case, it has been implemented a workflow for converting SuperStar files to DL3 format applied to the MAGIC project. Based on workflow modeling, a parallelism implementation strategy is proposed in the most important time stages of computing, improving performance with a speed of 3.14x in the critical stages of lower hard disk usage. Access to disk is a limiting factor that does not allow the improvement of other stages, solutions are proposed to overcome resource limitations.

Keywords– Reproducible, Data analysis, ReAna, Kubernetes, Docker, Containers, Rancher, Workflows, Common Workflow Language, DL3, MAGIC, Parallelism.

1 Introducció

La preservació i la reproductibilitat de les dades obtingudes en la recerca científica és un dels *hot topics* actuals en la gestió de les dades.

La capacitat que tenen els nous instruments digitals per generar dades de major precisió, fa que els volums a gestionar siguin molt elevats, en alguns casos, de l'ordre de PetaBytes per any. Aquesta gran quantitat de dades un cop analitzades produeix nous conjunts de dades, o datasets que requereixen ser preservades, igual que els entorns de producció (llibreries, paquets de software, etcètera) per poder-ne garantir la reproductibilitat i l'anàlisi.

En el 2012 Nature va publicar un estudi en el que revisava deu anys d'investigació, on els investigadors van trobar que 47 dels 53 treballs de recerca biomèdica sobre la investigació del càncer no eren reproduïbles, és a dir, no mostraven totes les dades, feien un ús inadequat de proves estadístiques i feien ús de reactius1[1].

És per això que han sorgit diverses preguntes que ara mateix estan en debat en la comunitat científica: "Com es gestionen els conjunts de dades analitzats?", "Com es preserven en el temps aquestes dades?", "Com es permet la compatibilitat i la preservació del software si els entorns de còmput varien?". És per aquest motiu que la comunitat científica cerca eines que permetin garantir que les dades científiques es tractin segons els principis Findable, Acces-

• E-mail de contacte: oriol.martinezac@e-campus.uab.cat
• Mención realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Eduardo Cesar Galobardes (acadèmic) i Jordi Delgado Mengual (PIC)
• Curs 2019/20

sible, Interoperable and Reusable[2] (FAIR). Un altre aspecte que també influeix en la recerca d'eines que permetin la reproductibilitat és la recerca oberta, que pretén assegurar l'eficiència de la recerca i maximitzar la producció de nous resultats.

La reproductibilitat computacional s'ha convertit en un tema molt important per igual tant per als investigadors de sistemes informàtics com els científics de domini. Tot i que la reproductibilitat computacional pot semblar més senzilla que replicar experiments físics, la complexitat i la rapidesa dels canvis en les tecnologies fa que la capacitat de reproduir l'anàlisi de dades sigui un gran repte.

Pel que fa a eines de preservació, estructuració d'anàlisi de dades científiques existeixen vàries eines:

- Computational Notebooks: Permeten definir fàcilment un anàlisi, estructurar-lo i documentar-lo. Són molt útils per a per a mostrar/ensenyar i són una forma convenient de compartir diferents anàlisis. Un dels notebooks més estesos en l'actualitat és Jupyter. Aquest permet crear python notebooks on programar i parametritzar l'anàlisi al mateix temps que es documenta cada passa. L'execució de Python notebooks vinculats als virtual environment de python permet tenir un sistema bastant complet.

- La comunitat de R també ha contribuït per a garantir que la investigació pugui ser reproduïble i transparent mitjançant el sorgiment d'un notebook per a R.2002, Sweave es va introduir en el 2002 per a permetre la incorporació de codi R en els documents de LaTeX per a generar fitxers PDF[3].

- Contenidors: Permeten empaquetar un software en un format que pot executar-se de forma aillada en un sistema operatiu compartit. A diferència de la màquina virtual, els contenidors no tenen un sistema operatiu corrent per a executar el software, només necessiten llibreries i configuracions necessàries perquè aquest funcioni. Això permet garantir que el software sempre s'executarà de la mateixa manera independentment de l'entorn en què es trobi[4]. Pel que fa a les tecnologies més exteses que utilitzen el sistema de contenidors hi ha Docker[5] i Singularity[6]. Ambdues tecnologies es poden utilitzar en l'execució de workflows en el REANA.

- REANA ofereix un conjunt d'eines per a modelar l'execució d'una anàlisi de dades, amb una visió integral de la descripció del worfklow a executar de manera que aquesta descripció permeti fer l'ànalisi reproduïble i reutilitzable en el temps. REANA suporta contenidors i específicament tant Docker com Singularity. Per afegir, REANA permet incorporar tanmbé descripcions fetes amb els notebooks i els virtual enviroments en les execucions dels workflows, això ofereix una alta flexibilitat pel que fa al software de base que s'utilitzarà per a realitzar l'anàlisi en aquest projecte.

En aquest treball s'explorarà l'eina REusable ANAlysis (REANA)[7], que permet estructurar, parametritzar i orquestrar l'anàlisi de dades de manera que sigui reproduïbles en el temps tot preservant el software. Aquesta eina reproduïble ha estat publicada pel Conseil Européen pour la Recherche Nucléaire (CERN). Per fer-ho s'utilitzarà com a exemple el workflow de conversió de dades a format Data Level 3(DL3)[8] que el projecte Major Atmospheric Gamma Imaging Cherenkov (MAGIC)[9] ha desenvolupat, per a la creació del seu *data legacy*.

REANA és una eina que està essent estudiada i provada en els camps de la física de partícules i l'astrofísica, i forma part d'un stack d'eines promogudes pel projecte ESCAPE (The European Science Cluster of Astronomy and Particle Physics of ESFRI research infrastructures) dintre de les convocatòries europees de recerca i innovació H2020.

Aquest projecte es realitzarà al Port d'informació Científica[10] (PIC). Un centre innovador de suport a la recerca ubicat a la Universitat Autònoma de Barcelona, que es dedica a través de l'ús de tecnologies de computació com Grid, High Throughput Computing o tecnologies de Big Data a oferir serveis com l'emmagatzematge i processament de grans quantitats de dades a col·laboracions científiques amb investigadors repartits arreu del món.

Aquest projecte s'inscriu en les tasques de recerca i innovació de la gestió i reprocessament de dades del projecte MAGIC, on el PIC gestiona el seu Data Center amb aproximadament 2 PetaBytes de dades.

A continuació d'aquest mateix apartat, s'introdueixen els elements més rellevants que han sigut cas d'estudi del treball. En primer lloc el framework REANA amb els seus components i seguidament el workflow d'estudi. A l'apartat de "Desenvolupament" es presenta el desenvolupament de la solució adoptada per tal d'implementar la plataforma que desplega REANA i el modelatge del workflow, així com un estudi preliminar de l'execució del workflow. A l'apartat "Resultats" es presenten els principals resultats obtinguts en les diferents propostes de millora de l'execució del workflow. Finalment a l'apartat "Conclusions" es presenten les principals conclusions extretes del present treball així com les línies de continuació futures.

1.1 Introducció a REANA

La plataforma REANA ofereix un conjunt d'eines per a modelar l'execució d'una anàlisi de dades, amb una visió integral de la descripció del workflow a executar de manera que aquesta descripció, permeti fer l'anàlisi reproduïble i reutilitzable en el temps. REANA permet als investigadors estructurar les seves dades d'entrada, codi d'anàlisi, entorns containeritzats i workflows computacionals per tal que l'anàlisi es pugui instanciar i executar en clouds de càlcul remots. REANA va néixer per orientar el cas d'ús de l'anàlisi de la física de partícules, però es pot aplicar a qualsevol disciplina científica. En aquest projecte s'estudiarà el seu funcionament i s'implementarà un cas d'ús amb un workflow de conversió de dades.

El sistema de REANA està format per diversos components majoritàriament desenvolupats amb Python que se simplifiquen en dues parts: el Gestor[11] de workflows i el Client[12] REANA. El Gestor és l'encarregat d'interpretar el workflow descrit i crea les tasques computacionals a executar. Aquestes tasques s'executen mitjançant Kubernetes[13] que automatitza la instanciació i l'escalat de l'aplicació en funció del nombre de peticions a executar per cada aplicació i usuaris. El Client de REANA és un entorn de Python, amb els paquets i funcionalitats cor-

responents que s'utilitzen per connectar i enviar peticions al Gestor per instanciar l'execució d'un workflow. Cal remarcar que el Gestor de REANA genera una carpeta on es carreguen tots els codis, inputs i outputs de l'execució d'un workflow, la qual cosa permet que el Client de REANA puguir recuperar aquests fitxers mitjançant l'ús de comandes. La plataforma de REANA també inclou un Client Web, tanmateix aquest es troba encara en fase de desenvolupament i no s'ha utilitzat en el desenvolupament d'aquest projecte.

El Gestor de REANA està format pels següents components, tal com es pot veure en l'esquema de la **Figura 1**:

- **REANA-Server**: Ofereix la funcionalitat necessària perquè els seus usuaris puguin reproduir l'anàlisi. La seva principal responsabilitat és l'autenticació i l'autorització de les peticions dels usuaris.
- **REANA-Workflow-Controller**: S'encarrega d'instanciar i gestionar els workflows computacionals. Oferint opcions com starting, stopping i deleting workflows.
- **REANA-DB**: Proveeix els models i utilitats de la base de dades.
- **REANA-Workflow-Engine**: S'encarrega d'executar els workflows computacionals. La versió 0.5.0 del REANA utilitzada en aquest treball suporta tres tipus de sintaxi descriptiva de workflows: Serial Workflow Engine, workflows seqüencials, CWL Engine, workflows de tipus Common Workflow Language[14] (CWL) i el Yadage Workflow Engine, workflows de tipus Yadage.
- **REANA-Job-Controller**: La seva responsabilitat principal és oferir una API comuna per carregar i gestionar treballs en múltiples backends de còmput com ara Kubernetes, HTCondor, SLURM, etc.

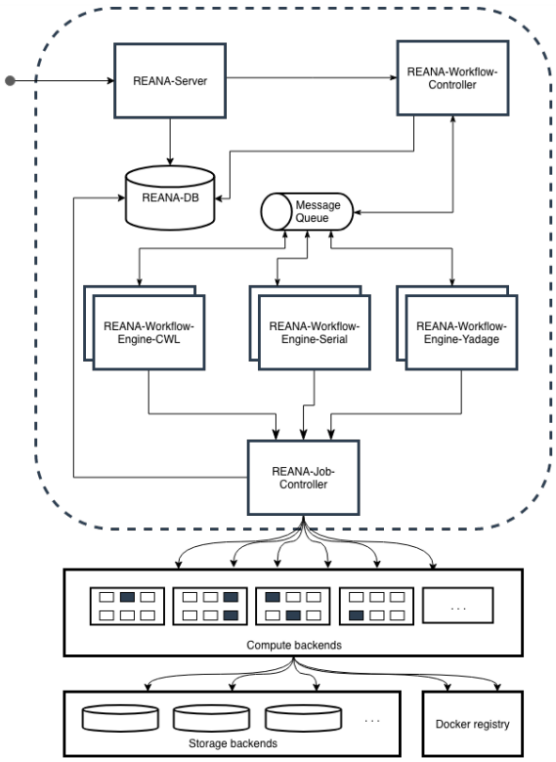


Fig. 1: Components de la plataforma de REANA.

Per a poder executar un workflow en el REANA és necessari definir prèviament el tipus de workflow a executar. El sistema suporta tres definicions de workflows: Common Workflow Language (CWL), Yadage o la implementació bàsica de workflows serials. En la descripció de l'execució cal definir també la ubicació de les dades d'entrada/sortida, les fases, l'entorn d'execució i les capes de connexió amb una infraestructura d'execució tipus Cloud o Cluster/Batch System.

De tots els tipus de workflow que podem executar en la plataforma REANA ens centrarem amb el CWL, ja que s'està esdevenint un estàndard i ens permet un nivell de paral·lelització de les fases del workflow que utilitzarà.

1.2 Introducció al Workflow de conversió a fitxers DL3

Les dades que s'utilitzen durant tota l'execució del workflow són dades generades a partir de les observacions en mode estereo dels dos telescopis Cherenkov MAGIC situats a l'Observatori del Roque de los Muchachos a l'illa de La Palma, Illes Canàries[15].

Les observacions es produeixen cada nit en cicles d'observació segons els paràmetres d'interès de la comunitat científica i en especial, la col·laboració internacional MAGIC que gestiona i opera l'instrument. Ambdós telescopis segueixen la mateixa planificació d'observació i operen de manera conjunta.

Cada telescopi està format per un gran mirall segmentat que enfoca la radiació de Cherenkov en una matriu de tubs fotomultiplicadors que formen una càmera. Els fotomultiplicadors estan acoblats a electrònica ràpida que amplifica, digitalitza i emmagatzema la imatge de la cascada de rajos gamma que impacten contra l'atmosfera de la Terra.

La calibració dels telescopis i el sistema de coordenades permeten definir una regió d'interès o camp de visió FOV (de l'anglès Field of View). Les imatges són emmagatzemades per separat en una estructura de directoris que permet separar les dades captades pel telescopi M1 i M2.

Aquestes imatges captades pels telescopis són l'equivalent a les dades crues (RAW Data) que poden contenir soroll i altres dades que no representen el senyal observat.

La RAW Data es processa per reconstruir el senyal detectat mitjançant el software propietari de MAGIC, anomenat MARS[16].

MARS és una suite d'eines que permet definir el processament de reconstrucció i l'anàlisi de dades, aquest software està implementat sobre l'API de ROOT[17]. ROOT és un conjunt d'eines de programari científic modular, que proporciona les funcionalitats necessàries per al tractament de dades, anàlisi estadística, visualització i emmagatzematge de grans quantitats de dades.

Un cop reconstruït el senyal de cada telescopi per separat, les deteccions capturades pels telescopis es combinen en un únic senyal. Aquestes dades són anomenades SuperStar.

El workflow que s'analitzarà en aquest treball s'encarrega de convertir les dades de SuperStar en dades de DL3, aquestes dades representen un format interoperable amb altres instruments Cherenkov com HESS, VERITAS o CTA, amb el qual comença l'anàlisi d'alt nivell per a interpretar l'observació realitzada.

Per a poder realitzar la conversió, cal definir unes fases intermèdies que permetin generar els principals components que descriuen el format DL3. Aquests són: una taula d'esdeveniments observats i les condicions en el moment d'observació i calibració de l'instrument que permeten interpretar i analitzar les dades, les anomenades Instrument Response Functions (IRFs).

Actualment aquest workflow es troba en estudi pel que fa a l'ajustament de paràmetres i qualitat dels fitxers DL3 obtinguts. És per això que les fases rebran diferents paràmetres de configuració d'entrada que vénen descrits pels fitxers anomenats input cards (fitxers amb extensió .rc).

A continuació s'expliquen breument cadascuna de les fases del workflow corresponent a la **Figura 2** que intervenen en la conversió de dades, així com les entrades i sortides de cada etapa[18, 19].

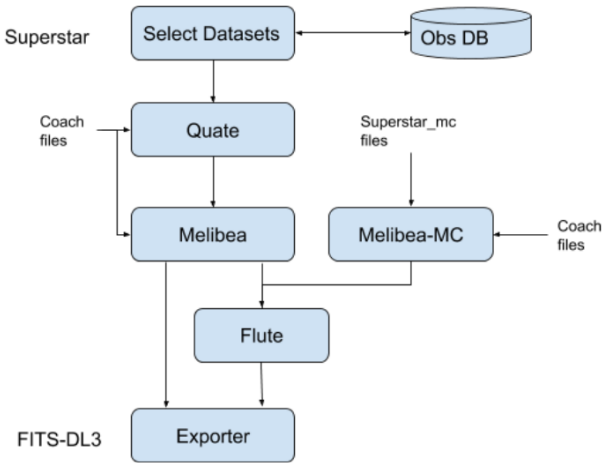


Fig. 2: Diagrama conceptual de les fases del workflow de conversió de dades DL3.

En les fases Quate, Melibea i Melibea-MC del workflow es rebran com a entrada els Coach files. Els Coach files són fitxers entrenats seguint el mètode Random Forest[20].

En la fase Melibea-MC s'utilitzaran com a entrada els fitxers Superstar_mc. Aquests fitxers s'obtenen a partir d'aplicar les simulacions Monte Carlo.

Dataset:

En la fase de selecció de dades es defineix una query a la base de dades d'observacions de MAGIC per obtenir un llistat de fitxers (Dataset, sortida) per a processar a partir d'uns criteris d'observacions (entrada).

- Entrada: Zenith-angle min i max (angle de visió), període d'observació, Date_min i Date_max (límits del període d'observació), el Wobble (selecció de la zona on es rep el senyal) i tipus de fitxer (indica la selecció de dades reals). Paràmetres addicionals, font (posició en el cel a observar), nit (data de l'observació) i No_moon (Observació sense filtres per llum de la lluna).
- Sortida: Llistat de fitxers Root per a processar.

Quate:

En la fase Quate es calcula les mitjanes d'un conjunt de paràmetres sobre les execucions i es realitza la selecció i classificació de les dades. En altres paraules, Quate determina si els fitxers d'entrada compleixen o no els paràmetres assignats, el més significatiu és el Zenith-angle.

- Entrada: Les dades seleccionades mitjançant el dataset els Coach files i un input card.
- Sortida: Els fitxers d'entrada classificats en quatre carpetes segons els paràmetres azimuth, zenith angle, bona qualitat i mala qualitat.

Melibea:

En la fase Melibea es converteixen els fitxers Superstar en una taula d'esdeveniments acompanyats de l'estimació de diverses propietats, energia estimada, hadronness i reconstrucció de la direcció de l'esdeveniment.

- Entrada: Un input card i els fitxers de sortida de bona qualitat generats pel filtratge del Quate. Els fitxers de sortida del Quate s'hauran de dividir per paquets. La granularitat de l'etapa Quate és una part de l'estudi del rendiment del workflow a executar.
- Sortida: Generarà una carpeta a partir de l'execució amb els nous fitxers en format Melibea.

Melibea-MC:

En la fase de Melibea-MC es processen les simulacions Superstar_mc per produïr les Melibea equivalents.

- Entrada: Dades simulades corresponents al Superstar_mc i els Coach files.
- Sortida: Dades simulades pel Melibea-MC.

Flute:

En la fase Flute es tracta de realitzar l'anàlisi del tractament del senyal contingut en les dades d'entrada en funció de paràmetres de configuració recollits en els inputs cards propis de cada etapa i atenent a la calibració de la càmera del telescopi. En aquesta fase s'obté l'espectre d'energia i la corba de llum d'una font de raigs gamma observada així com les IRFs, la instrumentació i condicions d'observació. Les IRFs són una part molt important dels continguts a escriure als fitxers DL3. Sense les IRFs, les dades en format DL3 no serien analitzables.

- Entrada: Fitxers de sortida generats pel Melibea i pel Melibea-MC. Aquesta etapa del workflow caldrà executar-la amb dos input cards diferents d'acord amb un tractament del senyal diferenciat (full-irf o point-like).
- Sortida: De tots els fitxers generats només es considerarà les IRFs calculades que caracteritzen l'instrument.

Exporter:

En la fase Exporter es creen els fitxers DL3 seguint les especificacions del format i combinant la taula d'esdeveniments generada a l'etapa Melibea amb les IRFs calculades pel Flute, utilitzant un format contenidor com és FITS[21].

- Entrada: Fitxers de sortida de la fase Flute.
- Sortida: Fitxer de tipus Data Level 3 que conté objectes amb taules EVENTS i IRFs.

2 DESENVOLUPAMENT

El desenvolupament d'aquest treball s'ha organitzat en tres parts. La instal·lació de la plataforma REANA, la modelització del workflow de conversió de dades a format DL3 i el posterior testeig i anàlisis de l'execució.

2.1 Implementació de REANA

Per afrontar la primera part d'aquest treball, ha calgut una anàlisi de la documentació del framework REANA, per entendre els components, la interrelació i el funcionament, per a poder instal·lar una instància completa del Gestor de workflows i del client REANA. En el transcurs d'aquest treball es farà servir la versió de REANA 0.5

Els recursos disponibles per a la implementació de REANA són una màquina virtual amb la distribució de Linux CentOS 7 i amb el següents recursos assignats: 5 CPUs (Intel Core i7 8700), 8GB de RAM (DDR4) i amb un disc dur Segate BarraCuda amb una velocitat lectura/escritura mitjana de 156 MB/s.

La instal·lació *out of the box* de REANA està preparada per una implementació senzilla i local de Kubernetes mitjançant Minikube. Pensant en un entorn de majors prestacions i escalabilitat (PIC) s'ha optat per fer una instal·lació i un desplegament de Kubernetes mitjançant Rancher[22], aquest ofereix una major escalabilitat i eines de gestió del cluster de Kubernetes tant pel que fa al control d'instàncies com per a la gestió de volums de disc, etc. Rancher és un sistema ja present i conegut al PIC per altres serveis.

Tot el stack de software de REANA, Rancher, Kubernetes i les tasques derivades de l'execució del workflow s'executen a la mateixa màquina. Com es detallarà més endavant, això presenta un problema de rendiment i limitació de recursos, però resulta suficient per provar tota la funcionalitat. La implementació amb Rancher permet la possibilitat d'escalar fàcilment en un futur a entorns amb més prestacions, com pot ser el Cloud o sistemes Batch com HTCondor.

Abans de tractar la implementació del workflow que descriu el cas d'ús d'aquest treball, s'han realitzat proves amb dos workflows simples de prova, que descriuen proves unitàries i de funcionalitat de l'entorn i del llenguatge de descripció CWL. Un primer workflow amb fases seqüencials i un altre amb fases paral·leles.

2.2 Implementació i modelització del workflow

Totes les etapes del workflow utilitzen MARS per a executar el processament, per instal·lar-lo s'ha generat una imatge de Docker que conté el software de MARS i els fitxers necessaris per a executar cadascuna de les fases del workflow. S'ha optat per una estructura modular i incremental per a definir la imatge de Docker, de manera que sigui més fàcil actualitzar cadascun dels seus components. L'estructura de Docker permet generar imatges a partir d'altres imatges de base, és per aquesta raó que la imatge generada ha sigut de forma modular i incremental. En la **Taula 1** es pot veure els mòduls (capes) que donen lloc a la imatge final mencionada.

TAULA 1: TAULA AMB ELS MÒDULS DE LA DOCKER IMAGE QUE S'EXECUTARÀ EN LES ETAPES DEL WORKFLOW.

Imatge base	CentOS 7.3
Imatge base + Root	Root 5.34.36
Imatge base + Root + Mars	Mars 2.19.9
Imatge base + Root + Mars + Inputs	Mars 2.19.9

Tal com es pot veure en la taula, si els inputs canviessin, només s'hauria de generar una nova imatge a partir de la imatge amb el software de Mars.

Dins de la imatge final hi haurà els fitxers d'entrada del workflow: input_cards, datasets, coach files i superstar_mc files. El motiu pel qual s'han inclòs tots aquests fitxers dins de la imatge i no com a punt de muntatge visible dins de la imatge de Docker, és perquè el REANA en la versió 0.5.0 de forma local no admet l'opció d'adjuntar un volum de disc addicional visible per a tots els PODs de Kubernetes. En la **Figura 3** es pot veure l'estructura interna de la unitat mínima d'execució de les fases del workflow dins del cluster de REANA.

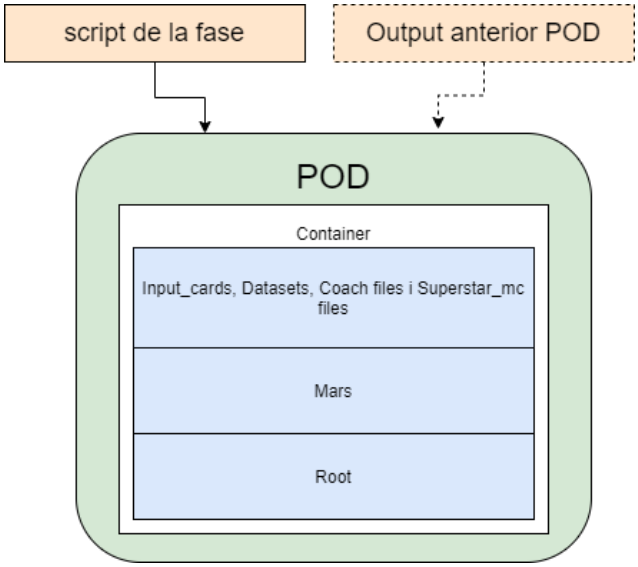


Fig. 3: Diagrama de l'estructura del Pod d'execució.

Com a entrada de cadascun dels PODs hi haurà: un ShellScript, que s'encarregarà de preparar i executar el software corresponent per aquella fase del workflow, i en el cas d'etapes intermèdies del workflow, tots els fitxers d'output generat pel POD predecessor. Cal remarcar que les subfases del Quate al ser els primers PODs a executar-se, no rebran com a entrada cap sortida de cap altre POD ('Output anterior POD' en la **figura 3**).

Un cop definit l'entorn d'execució de cadascuna de les fases del workflow, s'ha ampliat el diagrama conceptual del funcionament del workflow mostrat anteriorment per a tenir una idea més estructurada i detallada de l'execució del workflow dins de REANA. En la **Figura 4** es pot veure el diagrama detallat.

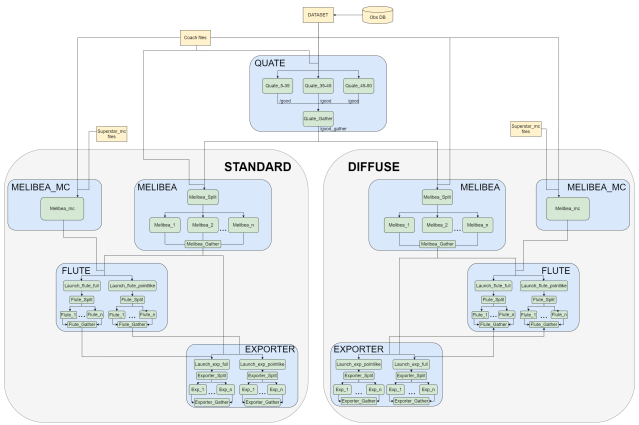


Fig. 4: Diagrama detallat del workflow.

Tal com es pot apreciar en el diagrama detallat hi ha dues parts, una pel que fa al processament 'Standard' i l'altre pel que fa al processament 'Diffuse'. Aquests tipus de processaments vénen definits per la naturalesa d'aquest workflow de conversió de dades. Representen modes de parametrització diferents que estan especificats en els inputs cards de cada etapa. Els modes es corresponen amb les dades MonteCarlo simulades que també intervenen en altres etapes del workflow i formen part tant de la descripció com del tractament de senyal del workflow.

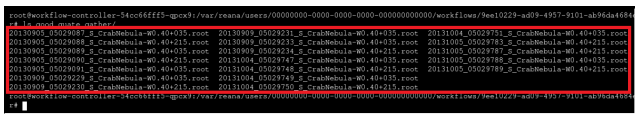
També es pot veure que cadascuna de les fases del workflow s'ha dividit en subfases. Aquestes són l'equivalent als PODs d'execució dins del cluster de REANA. En el cas de l'etapa de Quate aquesta se subdivideix en tres execucions que vénen definides pel rang d'angles zenitals que es fa servir com a paràmetre d'entrada. El rang complet va de 5º a 50º. Aquest rang per necessitat de l'estudi i l'execució se subdivideix en tres nous rangs: 5º-35, 35º-45º i 45º-50º. Aquesta subdivisió ve donada en funció del dataset i els paràmetres d'observació i, té la finalitat de cercar les dades de més bona qualitat amb major precisió.

En el cas de les etapes Flute i Exporter existeix un nivell addicional de paral·lelisme que també ve marcat pel disseny del propi workflow, on es situen les execucions amb els diferents input cards (full-irf i pointlike) en paral·lel doncs són completament independents.

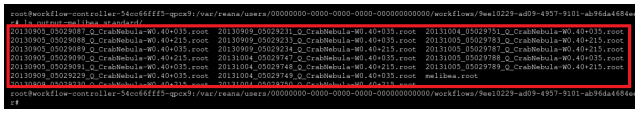
Pel que fa a Melibea, Flute i Exporter hi ha definit les subfases Split i Gather. L'Split s'encarrega de fer una distribució balancejada del conjunt de fitxers a processar segons la quantitat de nodes disponibles en l'execució paral·lela de l'etapa i optimitzant així el temps d'execució. La quantitat de nodes a instanciar en paral·lel de la fase del workflow és part d'estudi del rendiment que s'explicarà més endavant de forma detallada. Finalment el Gather s'encarrega de reunir tots els resultats generats per cadascun dels nodes.

Per a estudiar l'execució del workflow i tenint en compte la limitació de recursos disponibles, s'ha optat per l'execució de la branca Standard del workflow. L'altra branca, Diffuse, té exactament la mateixa estructura però variant els input cards a cada nivell.

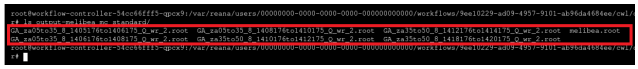
En la **Figura 5** es mostren els outputs de l'execució de cada etapa del workflow identificats per lletres: a) Quate, b) Melibea, c) Melibea_MC, d) Flute (full i pointlike) i e) Exporter (full i pointlike).



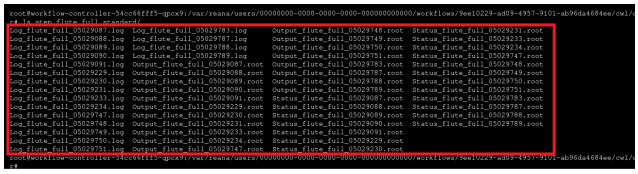
(a) Fitxers de sortida 'Good' de l'etapa Quate.



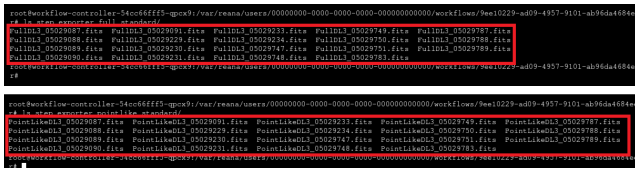
(b) Fitxers de sortida de l'etapa Melibea_standard.



(c) Fitxers de sortida de l'etapa Melibea_mc_standard.



(d) Fitxers de sortida de l'etapa Flute.



(e) Fitxers de sortida de l'etapa Exporter.

Fig. 5: Fitxers de sortida de cadascuna de les fases del workflow.

En la **Figura 5a** es mostra la sortida de realitzar una recol·lecció de tots els fitxers classificats per l'etapa Quate com a bona qualitat després d'executar les tres subfases quate_5-35, quate_35-45 i quate_45-50 amb els rangs de zenith corresponents. Com a entrada del dataset es reben 31 fitxers (.root) dels quals només 19 han estat classificats com a bona qualitat després d'aplicar els paràmetres definits en el input card. Tots els fitxers classificats per l'etapa Quate tenen el _S_ al nom, això fa referència als fitxers de SuperStar que són definits pel dataset inicial i seleccionat.

En la **Figura 5b** es poden veure els fitxers resultants d'executar l'etapa Melibea. En l'etapa de Melibea es generen el mateix nombre de fitxers de sortida que d'entrada. Aquests fitxers de sortida, s'identifiquen amb els caràcters _Q_ al nom. El fitxer melibea.root, és un fitxer de traçabilitat de l'execució que obviarem.

En la **Figura 5c** es poden veure els fitxers resultants d'executar l'etapa Melibea_MC. Per cada fitxer superstar MonteCarlo d'entrada es generaran tres fitxers.

En la **Figura 5d** es poden veure els fitxers resultants d'executar l'etapa Flute (full i pointlike). Per a cada fitxer d'entrada de Melibea es generaran tres fitxers de sortida: Output_*.root, Status_*.root i un log (Log_*.Log). Els fitxers amb el nom Status són fitxers gràfics que poden obrir-se a través d'un visor. Es pot verificar que hi ha 19 fitxers de cada tipus.

En la **Figura 5e** es poden veure els fitxers resultants d'executar l'etapa Exporter (full i pointlike). En aquesta etapa el nombre de fitxers de sortida serà el mateix que els fitxers classificats com a bona qualitat en l'etapa Quate, és a dir, 19 fitxers de tipus FITS.

2.3 Estudi de l'execució del workflow

Un cop realitzada la implementació de la funcionalitat de la branca 'Standard' del workflow s'ha capturat els temps d'execució de cadascuna de les fases en una execució serial, per tal de processar el nombre de fitxers d'entrada corresponents a cada etapa. En la **Figura 6** es mostra un graf de precedències de les fases del workflow (Standard) executat al cluster de REANA amb els seus respectius temps d'execució. Aquest temps s'ha obtingut a partir de fer una mitjana dels temps equivalents a 3 execucions del workflow.

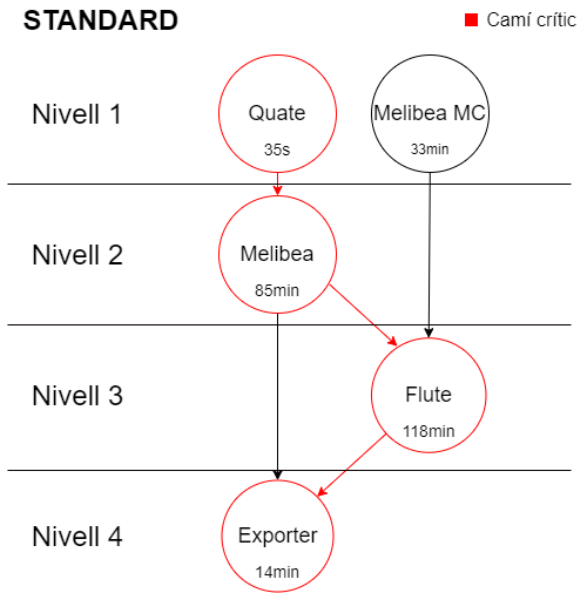


Fig. 6: Graf de precedència de les fases del workflow (Standard) i el seu temps d'execució.

Tal com es pot observar en el graf les etapes de Melibea, Flute i Exporter són part del camí crític de l'execució del workflow.

A continuació es mostra en la **Figura 7**, una gràfica amb l'impacte (percentatge) de cadascuna de les fases respecte al temps total d'execució del workflow. D'aquesta manera podem quantificar i per la qual cosa centrar-nos a millorar els temps d'aquelles etapes que tinguin un major percentatge en el temps total. Com es pot veure, Flute, amb un 54.23%, i Melibea, amb un 39.07%, són les etapes que més triguen a executar-se.

Segons la definició del workflow i del dataset a processar, es pot observar que cada etapa té un conjunt de N fitxers a processar. Cada execució de les fases Flute o Melibea són independents entre si, de manera que els N fitxers d'entrada poden reorganitzar-se. La proposta que es fa és passar d'una execució seqüencial de N fitxers, a una execució paral·lela.

Aquesta proposta implica executar en paral·lel tantes instàncies, n_instàncies, de cada fase com siguin possibles, tenint en compte, la relació entre el nombre de fitxers d'entrada i el temps d'execució de la fase en processar els N fitxers.

S'enfocarà en fer una execució parcial de workflow centrant-se en l'optimització de les fases Flute, Melibea i Exporter atès que formen part del camí crític, i per tant, amb millores en aquestes etapes es pot assolir una millora global de l'execució del workflow.

Per a implementar el paral·lelisme a les execucions dels N fitxers d'entrada a cadascuna de les etapes, s'ha introduït un pas previ a l'execució de cada etapa anomenat "Split". Aquest organitza i distribueix la càrrega de fitxers a processar segons un paràmetre d'entrada que indica el nombre d'instàncies a executar. D'aquesta manera cada instància rep M=N/n_instàncies fitxers d'entrada, per a tenir una càrrega el més balancejada possible.

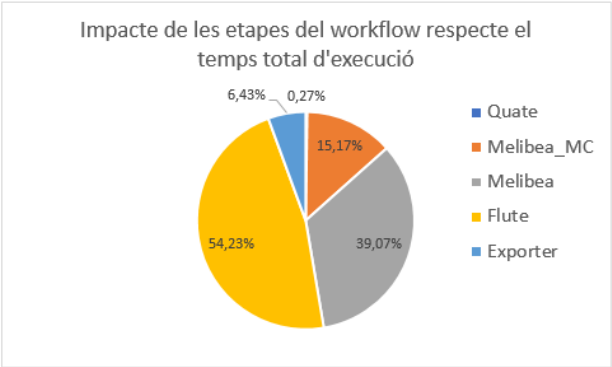


Fig. 7: Impacte en % de cadascuna de les etapes del workflow respecte el temps total d'execució.

Durant l'estudi del rendiment del workflow aplicant paral·lelisme han sorgit problemes, ja que el processament del workflow es troba en el mateix node que la gestió del cluster de Kubernetes. Les fases Melibea, Melibea_MC, Flute i Exporter realitzen un elevat nombre d'escriptures i lectures simultànies a disc que provoca una caiguda de rendiment del sistema. En l'apartat "Resultats" es mostrarà l'impacte sobre el disc dur en l'execució de cada etapa.

Per aquesta raó s'ha optat per fer un estudi de millores de rendiment parcials, és a dir, executar només una fase del workflow en concret i mirar el temps obtingut en relació amb les instàncies en paral·lel creades. D'aquesta manera tot i no poder fer una optimització completa ateses les limitacions hardware de les que es disposen, sí que permet identificar les estratègies de paral·lelització a aplicar per la millora de cada etapa, i així, extrapolar a l'execució global del workflow utilitzant REANA.

3 Resultats

En aquest apartat s'exposaran i s'interpretaran els principals resultats obtinguts al llarg del treball. D'una banda, la valoració de la implementació i funcionalitat de REANA i de l'altre, els resultats d'implementar estratègies d'optimització basades en el paral·lelisme de les etapes del workflow del camí crític.

Com a primers resultats a valorar des d'un punt de vista més funcional, destacar que s'ha instal·lat REANA mitjançant l'orquestrador de Kubernetes a través de Rancher. L'ús de Rancher a més, ha permès entendre tot el procés d'instanciació de les tasques d'un workflow, la ges-

tió de la comunicació entre PODS i tenir un orquestrador de Kubernetes que pugui escalar fàcilment amb més recursos hardware a l'abast.

Un cop REANA ha estat degudament instal·lat i provat amb diferents execucions sèrie o paral·leles d'exemple, s'ha pogut passar a la modelització d'una de les branques del workflow de conversió de fitxers DL3, Standard, que era objectiu d'aquest projecte. La modelització del workflow no ha estat completa per manca de temps, però resulta suficient per a la valoració del correcte funcionament, orquestració i gestió de les dependències de dades entre etapes, com s'ha mostrat a l'apartat de "Desenvolupament" d'aquest treball.

Per tal de valorar els resultats d'aplicar les estratègies d'optimització parcials d'algunes etapes s'utilitzarà la mètrica definida per l'Speed-Up. L'Speed-Up es defineix per la següent relació:

Speed-Up = temps execució en sèrie / temps execució etapa en paral·lel

En una aproximació teòrica ideal, l'Speed-Up d'execució en paral·lel hauria de seguir un creixement lineal, a mesura que s'assignen més nodes a l'execució. Un Speed-Up d'1 o inferior, no suposa cap millora en l'execució, en canvi, un Speed-Up superior a 1 aporta una millora de la nova execució versus la serial.

En la **Taula 2** es mostra el temps, en segons, d'haver executat les etapes Melibea, Flute i Exporter de manera seqüencial i paral·lela. El nombre de nodes especifica si l'etapa ha sigut executada de forma seqüencial o paral·lela. Un node és l'equivalent a executar l'etapa de forma seqüencial i més d'1 node és l'equivalent a executar l'etapa de forma paral·lela.

TAULA 2: TAULA AMB ELS TEMPS D'EXECUCIÓ DE LES ETAPES PARAL·LELITZABLES DEL WORKFLOW.

	Etapes		
Nodes	Melibea	Flute	Exporter
1	5191s	6669s	895s
2	4165s	3515s	574s
3	3514s	2118s	559s
4	3621s	2173s	541s

En la **Figura 8** es mostra una gràfica amb la mètrica Speed-Up de cadascuna de les fases a través d'augmentar el nombre d'instàncies (nodes d'execució).

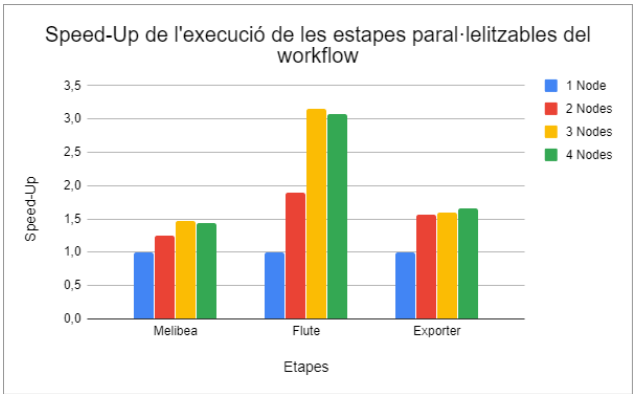
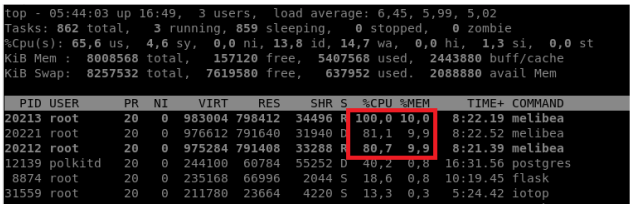


Fig. 8: Speed-Up de l'execució de les etapes paral·lelitzables del workflow.

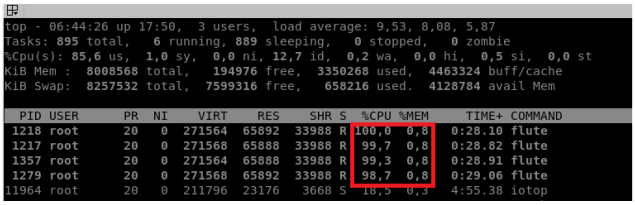
Com es pot veure en la gràfica de la **Figura 8**, les etapes Melibea i Exporter no presenten cap millora significativa del Speed-Up a l'hora d'augmentar el nombre d'instàncies en paral·lel, el màxim de Speed-Up que presenten és 1,48 i 1,65 respectivament a l'utilitzar 3 instàncies, mentre que la progressió que s'esperaria idealment seria lineal amb Speed-Ups de 2 i 3 a l'utilitzar 2 i 3 instàncies respectivament. En canvi Flute, amb el mateix nombre d'instàncies, si que mostra la millora esperada amb un Speed-Up de 3,14 amb 3 instàncies.

També es pot apreciar a la gràfica que en cap cas l'ús de 4 instàncies amb els recursos a l'abast suposa una millora de rendiment.

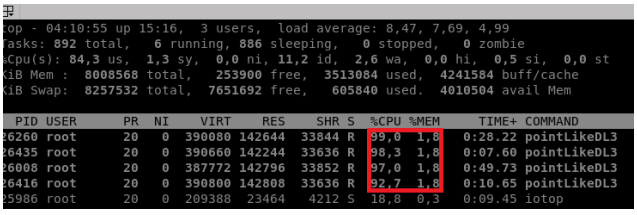
Per a interpretar la causa de la manca d'escalat en assignar més recursos, s'ha fet un *profiling* de les execucions i s'ha extret informació del monitoratge del sistema. En aquest s'ha pogut observar l'ús de CPU i RAM de cadascuna de les instàncies de cada etapa: Melibea utilitza aproximadament un 87.27% de CPU i un 9.9% de memòria, Flute utilitza aproximadament un 99.43% i un 0.8% de memòria i Exporter utilitza aproximadament un 96.75% i un 1.8% de memòria. En la **Figura 9** es poden veure els percentatges d'ús de CPU i RAM en les diferents etapes paral·lelitzades a través de la comanda top.



(a) Output comanda top de Melibea.



(b) Output comanda top de Flute.



(c) Output comanda top d'Exporter.

Fig. 9: Monitoratge de la comanda top de les fases Melibea, Flute i Exporter.

Com es pot veure en la **Figura 9**, la RAM no presenta una dificultat, en canvi, la CPU mostra una càrrega elevada per a cadascun dels processos de les instàncies de les etapes. A la **Figura 10** es pot veure que amb els recursos assignats a REANA (5 CPUs), l'ús de la CPU no representa un problema en el rendiment per a les execucions de fins a 4 instàncies. Tanmateix, en el moment que el nombre d'instàncies de les etapes paral·lelitzables sigui superior a les CPUs assignades, això pot mostrar un impacte negatiu en el rendiment de la seva execució.

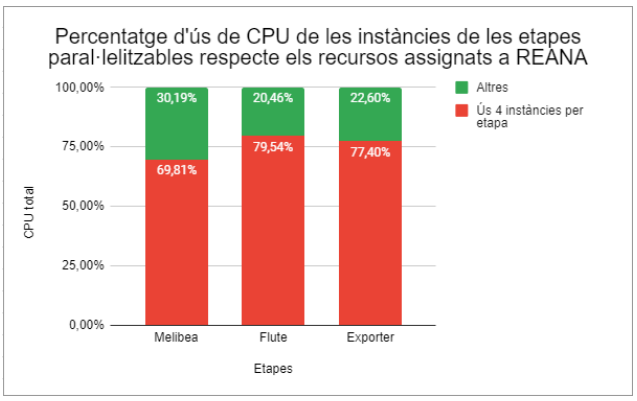
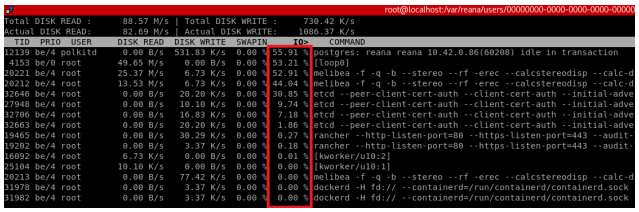
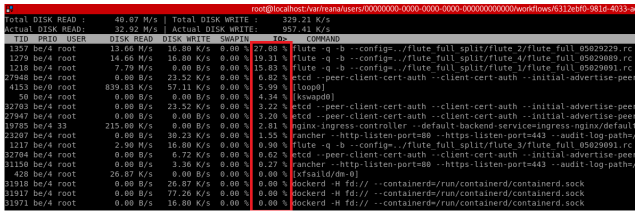


Fig. 10: Percentatge d'ús de CPU de les instàncies de les etapes paral·lelitzables respecte els recursos assignats a REANA.

Pel que fa al disc, s'ha pogut veure un elevat percentatge d'ús en les instàncies de les fases Melibea i Flute, aproximadament un 49% i un 20% respectivament, i també a causa de la gestió de REANA. A través de la **Figura 11** es pot veure el percentatge d'ús del disc en les diferents etapes paral·lelitzades. Cadascuna de les instàncies d'una mateixa etapa que volen llegir/escriure al disc dur hauran d'accedir de forma seqüencial, per la qual cosa es pot concloure que el Bottleneck de l'execució del workflow és el disc dur.



(a) Output comanda iotop de Melibea.



(b) Output comanda iotop de Flute.

(c) Output comanda iotop d'Exporter.

Fig. 11: Monitoratge de la comanda iotop de les fases Melibea, Flute i Exporter.

4 Conclusions

A continuació es presenten les principals conclusions que s'extreuen del treball realitzat.

Aquest treball ha servit per a crear un prototip de plataforma d'anàlisis de dades reutilitzables i preservables en el futur mitjançant REANA, amb un cas d'ús per al projecte MAGIC. La implementació de la infraestructura, tot i que amb recursos limitats, ha sigut satisfactòria per a entendre el seu funcionament i per a entendre la implementació del workflow de conversió de dades en el framework REANA. El workflow es divideix en dues grans branques, les quals tenen la mateixa estructura i dependències, i varien en els paràmetres d'entrada mitjançant els input cards. Per manca de temps, només s'ha implementat la branca de processament Standard, però les estratègies de paral·lelització i resultats obtinguts són extrapolables a l'execució de la branca Diffuse.

El següent objectiu, un cop aplicat la funcionalitat del workflow, ha sigut estudiar el rendiment de l'execució del workflow i mirar d'implementar mecanismes d'optimització aplicant una estratègia de paral·lelisme a nivell de dades. Aquest objectiu s'ha assolit en part. Les estratègies d'aplicar paral·lelisme són parcials a l'execució completa del workflow, és a dir, s'han separat les etapes que presenten un patró paral·lelitzable i s'ha estudiat la millora de temps d'aquestes a través d'aplicar paral·lelisme. Els resultats obtinguts en l'aplicació de paral·lelisme a les etapes Melibea i Exporter no poden considerar-se com a millores o optimitzacions concloents de l'execució del workflow, ateses les limitacions de hardware, en especial l'accés al disc dur, però si serveixen com a una primera idea de l'estratègia per a implementar millores en execucions basades en el paral·lelisme. Com es pot observar a l'etapa Flute el rendiment és més proper a l'esperat atès que té un menor ús de l'accés a disc. Per tal de millorar l'execució i resoldre els problemes observats de rendiment vinculats a l'accés de disc i poder tenir un estudi de rendiment més acurat, caldria garantir que cada tasca del workflow s'executa en un entorn completament independent de l'orquestrador de REANA.

Un cop es garanteixi l'execució independent de cada tasca mitjançant una integració de REANA amb un servei de cloud o batch system, es podria fer una revaloració del rendiment i *profiling* de cadascuna de les etapes i generar un nou graf de precedència amb els nous resultats. A partir d'aquests nous resultats s'hauria de mirar si el camí crític obtingut segueix sent el mateix o ha canviat, i quines etapes són les que tenen un major impacte en el temps de l'execució del workflow. La contínua repetició d'aquest procés d'anàlisi seria l'equivalent a la feina realitzada per un enginyer d'anàlisi de rendiment.

4.1 Línies futures

En aquest apartat es mostren els següents objectius en cas de seguir amb el desenvolupament de l'estat actual del projecte.

- Tenir el volum de disc visible a tots els PODs seria clau, ja que la limitació de no poder muntar un volum de dades addicional i sincronitzat amb un volum de disc extern a les imatges de Docker i dels PODs, dificulta que la imatge Docker pugui ser agnòstica pel que fa a les dades a processar. Actualment cada vegada que es vol generar un nou dataset per a executar el workflow requereix regenerar la imatge. És per aquest motiu que, a més a més, s'ha decidit reduir la quantitat de fitxers del dataset que hi haurà en la imatge per evitar llargs temps de 'overhead'.

- Implementar la funcionalitat de la branca Diffuse i fer un estudi de l'impacte del paral·lelisme en el seu rendiment en comparació a la branca Standard que ha sigut cas d'estudi d'aquest treball. La implementació d'ambdues branques també portaria a una anàlisi de l'orquestració de l'execució d'ambdues branques en paral·lel.

- Actualitzar la versió del REANA a la versió 0.6.0. En aquesta versió de REANA s'ha afegit la compatibilitat amb sistemes de batch de còmput com HTCondor i Slurm per als workflow engines (Serial, CWL i Yadage)[23]. Aquesta compatibilitat permetria connectar la plataforma actual de REANA al cluster HTCondor del PIC. D'aquesta manera, es podria garantir que cada tasca generada pel workflow, sigui sèrie o paral·lela, s'executi en un entorn independent sense col·lisions a l'hora d'accedir als recursos hardware. Aquesta proposta permetria implementar noves estratègies per a millorar el rendiment de l'execució del workflow.

5 REFERÈNCIES

[1] REANA: Research and innovation, https://ec.europa.eu/info/sites/info/files/research_and_innovation/reana.pdf

[2] Findable, Accessible, Interoperable and Reusable: FAIR Principles - GO FAIR, https://www.go-fair.org/fair-principles/

[3] Reproducible-analysis-workshop.readthedocs.io: Tools and Platforms for Reproducibility and Transparency in research — Reproducible Research workshop 1.0 documentation, https://reproducible-analysis-workshop.readthedocs.io/en/latest/3.Tools-and-Platforms.html

[4] Reproducible-analysis-workshop.readthedocs.io: Docker and Reproducibility — Reproducible Research workshop 1.0 documentation, https://reproducible-analysis-workshop.readthedocs.io/en/latest/8.Intro-Docker.html

[5] Docker, What is a Container? — Docker, https://www.docker.com/resources/what-container

[6] Singularity, Singularity — Singularity, https://singularity.lbl.gov/

[7] REANA: REANA - Reusable Analyses, http://reanahub.io/

[8] DL3: Toward a Public MAGIC Gamma-Ray Telescope Legacy Data Portal, https://arxiv.org/pdf/1909.01172.pdf

[9] MAGIC: MAGIC data center @PIC, http://magic.pic.es/

[10] Port d'Informació Científica: PIC, https://www.pic.es/

[11] REANA: Reproducible research data analysis platform, Administrator Guide, https://reana.readthedocs.io/en/latest/administratorguide.html

[12] REANA: Reproducible research data analysis platform, User Guide, https://reana.readthedocs.io/en/latest/userguide.html

[13] Kubernetes: What is Kubernetes, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[14] Wikipedia: Common Workflow Language, https://en.wikipedia.org/wiki/Common_Workflow_Language/

[15] Wikipedia: Telescopi Cherenkov, https://ca.wikipedia.org/wiki/Telescopi_Txerenkov

[16] MARS: the MAGIC Analysis and Reconstruction Software, https://arxiv.org/abs/0907.0943

[17] ROOT a Data analysis Framework — ROOT a Data analysis Framework, https://root.cern.ch/

[18] Workflow: Indico.cern.ch, https://indico.cern.ch/event/783425/contributions/\3328248/attachments/1814935/2965931/presentation.pdf

[19] Data formats for gamma-ray astronomy: Data formats for gamma-ray astronomy 0.3.dev documentation, https://gamma-astro-data-formats.readthedocs.io/en/latest/index.html

[20] Towards Data Science: Understanding Random Forest, https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[21] FITS: FITS Documentation Page, https://fits.gsfc.nasa.gov/fits_documentation.html

[22] Rancher: Rancher Labs, https://rancher.com/what-is-rancher/overview/

[23] Reana.readthedocs.io: Changes — reana documentation, https://reana.readthedocs.io/en/latest/changes.html